SafeDrive: A Robust Lane Tracking System for Autonomous and Assisted Driving Under Limited Visibility

Jiawei Mo[1] and Junaed Sattar[2]

***Abstract*— Autonomous detection of lane markers improves road safety, and purely visual tracking is desirable for widespread vehicle compatibility and reducing sensor intrusion, cost, and energy consumption. However, visual approaches are often ineffective because of a number of factors, including but not limited to occlusion, poor weather conditions, and paint wear-off. We present an approach towards robust lane tracking for assisted and autonomous driving, particularly under poor visibility. Our method, named SafeDrive, attempts to improve visual lane detection approaches in drastically degraded visual conditions without relying on additional active sensors other than vision and location data, sensors that are readily available on a standard smartphone. In situations where lane markers are not visible, the proposed approach uses location information of the vehicle to access alternate imagery of the road at that particular location from an existing database of such images. These alternate images are subsequently used to reconstruct a sparse 3D model of the surroundings at the current location. By estimating the geometric relationship between this 3D model and the current view, the lane markers in the 3D model are projected onto the current view. We demonstrate the effectiveness of our system on actual driving data recorded in downtown Minneapolis of United States.**

***Index Terms*— Lane detection, vehicle safety, mobile computer vision.**

I. Introduction

Recent advances in affordable sensing and computing technologies have given new impetus towards commercialization of a wide variety of intelligent technologies. A major consumer-targeted application has focused on increasing autonomy in transportation systems, the most prominent of which is the area of self-driven cars. Autonomous driving has been a key focus in both academic and industrial research and development activities [20] as of late. Alongside fully autonomous commercial vehicles, mainstream auto manufacturers are equipping their vehicles with more intelligent technology with semi-autonomous, *assistive* features – the primary focus being increased safety. Many recent consumer-grade vehicles come with a number of such safety-enhancing features – *e.g.*, lane assist, blind-spot detection, radar-assisted braking, visual collision avoidance, driver fatigue detection [21] – with the number and quality of features increasing in higher-end, more expensive vehicles. These features are also available as add-on options, often costing a few thousand US dollars to install in a vehicle. Even then, not all vehicles are capable of fitting such a system, as these options require specific vehicle data and power interfaces, limiting their application to newer vehicles. However, to minimize distracted driving (which has approximately 20 *per cent* contribution to fatalities on the road [1]) and improve safety, many consumers are opting to buy newer vehicles with these features pre-installed.

The authors are with the Department of Computer Science, University of Minnesota, 200 Union St SE, Minneapolis, MN, 55455, USA {[1]moxxx066, [2]junaed} at umn.edu.

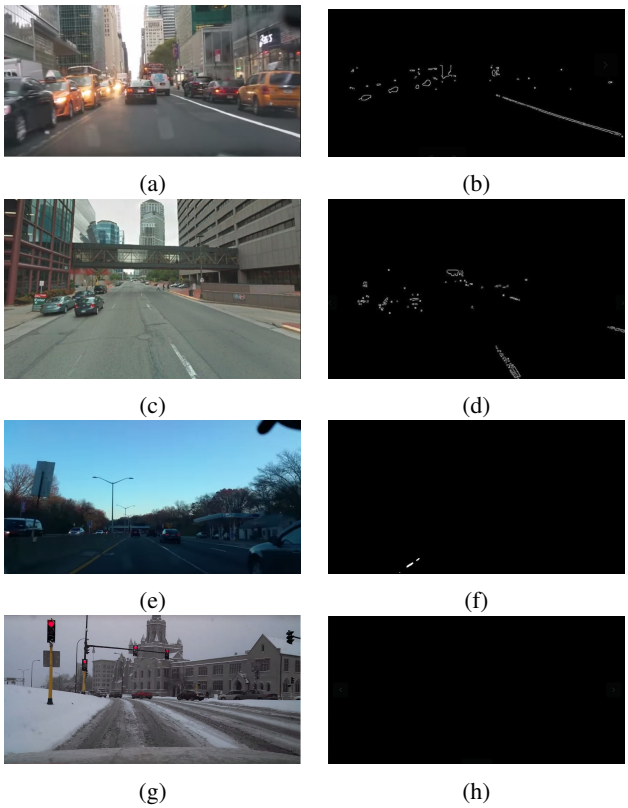


Fig. 1: Visual lane tracking on several urban scenes from YouTube™ videos. Snapshot (1a) (output in (1b)): lane markers not distinct in the center, though side markers are detectable. Snapshot (1c) (output in (1d)): lane markers mostly washed out. Snapshot (1e) (output in (1f)): evening drive, low-light conditions make the lane markers almost undetectable. Snapshot (1g) (output in (1h)): snow-covered roads, no lane markers visible.

Across manufacturers (and in some cases, vehicle models), a variety of sensing methods are used in order to provide accurate detection of road features and consequently prevent traffic mishaps. Such sensors include but are not limited to laser scanners, radar, proximity sensors, and visible-spectrum cameras. Vision is an unobtrusive, low-cost, low-power sensor but requires appropriate lighting, unobstructed view, and fast processing time for deployment in autonomous

and assisted driving applications. In spite of its disadvantages, vision sensors carry a strong appeal for deployment in mass production systems, mostly because of its low-power, inexpensive nature. For example, Subaru offers a stereo vision system termed EyeSight [2] for lane detection and collision avoidance, with a stereo camera pair mounted on either side of the center rear-view mirror. This provides depth perception and lane tracking, and an intelligent drive system provides adaptive braking and cruise control, collision avoidance and lane-departure warnings. While the system has shown to work well in manufacturer testing, it is not immune to common failure cases, namely occlusion of road surfaces from snow, mud or a large vehicle, poor lighting condition, variable weather and ambiguity arising from feature confusion. An example scenario is shown in Figure 1, where a sequence of four snapshots demonstrate how changing conditions affect the quality of the center lane markers in the visual scene. The lane detection system is using a real-time segmentation approach; however, irrespective of the particular algorithm used to segment or detect the center lines, the input images themselves are of significantly degraded quality for robust lane tracking.

This paper proposes a system called SafeDrive[1], which is a significantly inexpensive approach for robust visual lane detection in severely degraded conditions, without relying on exotic, costly sensors which would be prohibitive for financial and compatibility reasons. Under poor visibility, the system uses vehicle's location data to locate alternate images of the road from an available "road-view" database. A sparse 3D street model is reconstructed from the alternate images. Subsequently, lane markers in the 3D model are projected onto current view, according to the geometric relationship between current view and 3D street model. For development of SafeDrive, an Android-based application called DriveData has been developed to capture a variety of data from a device mounted on (or even outside) the vehicle. Our long-term goal is to provide an affordable solution to be used on a smartphone mounted on the windshield to provide lane departure warnings in extreme cases, and safety recommendations under the current driving conditions based on visual, location and acceleration data.

II. Related Work

A large body of literature exists on different aspects of autonomous driving and driver's assistance technologies, a number of which relies on robotics and computer vision methods. The Carnegie-Mellon NavLab [19] project has produced some of the earliest implementations of self-driving cars, and have extensively used vision for a number of subtasks, including road and lane detection [13]. Stereo vision systems have been used for lane detection; *e.g.*, in [14], [3]. Dedicated parallel processing for road and lane detection have been investigated in the GOLD [3] framework. Kluge et al. [12] applied deformable shapes and Wang et al. [23] used "snakes" for detection and tracking of lane markers. Spline-fitting methods for lane tracking have also been applied [22]. Kim [11] investigated robust lane tracking under challenging conditions with poor visibility and rapidly changing road traffic – similar in nature to the problem we are addressing in this paper. However, that work does not consider the case of zero-visibility of lane markers, which we attempt to resolve. We use color-based segmentation and line-fitting in our work, and a number of authors have investigated similar approaches (*e.g.*, [6], [7], [4]). Often used with a combination Bayesian filtering and estimation methods, these methods have shown to work well under clear visibility conditions. Some researchers looked into the problem of rear-view lane detection [18]. Interested readers are pointed to the paper by Hillel et al. [10] for an in-depth survey of recent advances in road and lane detection problems.

[1] https://github.umn.edu/moxxx066/SafeDrive

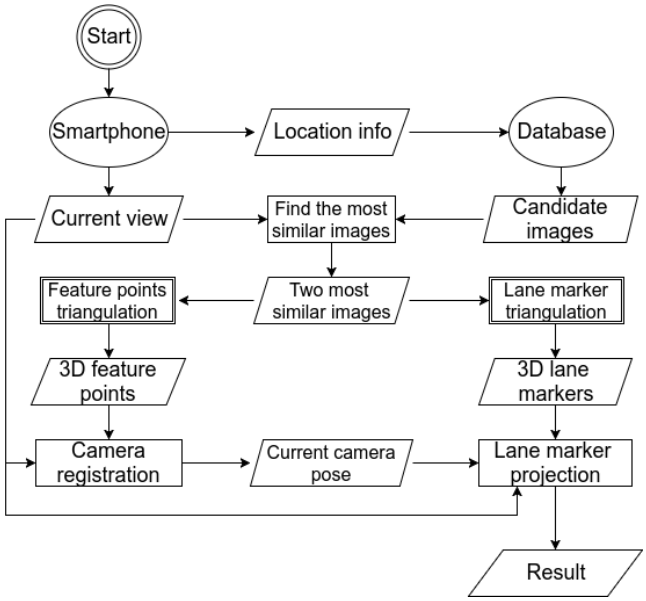


Fig. 2: Diagram depicting various stages of SafeDrive.

III. Methodology

The main idea behind SafeDrive is to detect lanes in *alternate imagery* of current location, which is presumably of better visual quality, and then project the detected lane markers into current image. This projection is done using geometric correspondences between current view and a sparse reconstructed view of current scene from alternate imagery, using a similar approach to Structure from Motion[9]. A database is searched to find images most similar to current scene, and a 3D model of street is reconstructed from these images. The current view is then registered into this 3D model. Eventually, the lane markers in the 3D model are projected onto current view. The entire process is illustrated in Figure 2.

A. Find the most similar images

In order to get the alternate images for 3D reconstruction, we search for the most similar images in database. For

search optimization, we index images in the database by their location, to improve search accuracy, as well as to reduce the number of possible images to be matched. For our purposes, the criterion we use to measure *similarity* between two images is based on the number of matched feature points. This is because we are not after true similarity, as the current and candidate images may have very different appearances. The intent is to find two images with maximally overlapped visual content, which will essentially ensure the highest number of feature point matches for realistic scenes. Feature points are detected using Harris corner detector[8]. Instead of keeping every feature point, we force the final feature point to be at least a certain distance away from any other feature point. This is to ensure an even spatial distribution of feature points to improve the likelihood of finding the most accurate image match. ORB feature descriptor [17] is used to extract feature descriptors from both current image and candidate image, and match between them. To ensure an accurate match, descriptor matching is run in both direction between current image and candidate image to further remove inconsistent matches.

After matching features on all possible candidate images at the current location, two images with the most-matched points are chosen for 3D reconstruction. Two images are the minimum requirement for 3D reconstruction. A higher number of images improves the quality of the 3D street model; however, as our focus is not on the 3D reconstruction itself, we use two images to reduce the computation load. While such a choice may question the resulting accuracy of the entire process, our experiments demonstrate that using two images for the reconstruction process is sufficient for SafeDrive if these images are not taken from geographically proximal locations. Future work could be done to eliminate this assumption, by refining the 3D model using more images.

B. Feature Points Triangulation

The process of feature points triangulation is illustrated in Figure 3.

Since we are only interested in the geometric relationship between the 3D street model and current view, instead of full 3D reconstruction, we build a sparse 3D street model with feature points only, which is sufficient for camera registration. To generate the 3D points needed for sparse 3D street model reconstruction, we need to estimate the relative pose between the two images. We match features between the image pair, then the correspondences are used to calculate the fundamental matrix between the two images. Subsequently, the rotation matrix and translation vector between the image pair are recovered from the fundamental matrix and camera intrinsic parameters. The translation vector is up to scale, but it is not a problem because: assume the scale is λ, then the generated 3D point is $\lambda(x, y, z)$, where (x, y, z) is the real position of this point in world coordinate. When registering the current view, the current camera pose will be $[\mathbf{R}, \lambda\mathbf{t}]$, compared to the real current camera pose $[\mathbf{R}, \mathbf{t}]$, where $\mathbf{R}$ is rotation matrix and $\mathbf{t}$ is translation vector. As a result, the transformed point in current camera frame before projection will be: $[\mathbf{R}, \lambda\mathbf{t}][\lambda x, \lambda y, \lambda z, 1]^T = \lambda[\mathbf{R}, \mathbf{t}][x, y, z, 1]^T$. Since we are using pinhole camera model, the scale factor is eventually eliminated.

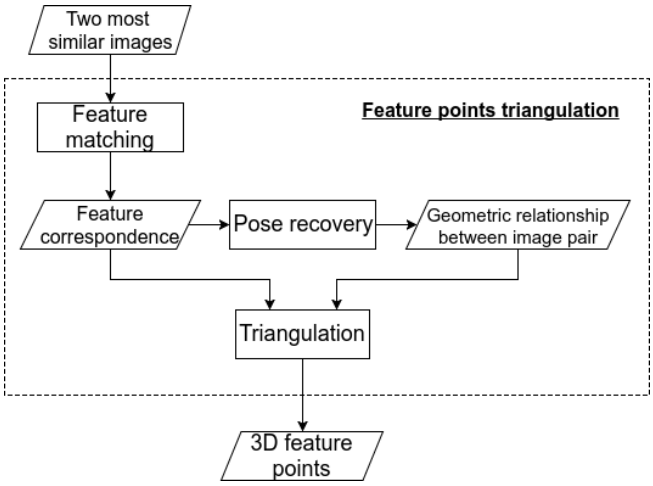


Fig. 3: Diagram depicting feature points triangulation in SafeDrive.

Based on the rotation matrix and translation vector, 2D feature correspondences are triangulated to get 3D points in the following manner:

We are going to solve for each 3D feature point X. We use K to represent the camera intrinsic parameters, and $[R_i, t_i]$ to represent the pose of the ith camera in world coordinate. For feature u_1 in the first image:

$$\begin{aligned} \lambda_1 u_1 &= K[R_1, t_1]X \\ u_1 \times \lambda u_1 &= u_1 \times K[R_1, t_1]X \\ 0 &= [u_1 \times K[R_1, t_1]]X \end{aligned}$$

Similarly for its corresponding u_2 in the second image:

$$0 = [u_2 \times K[R_2, t_2]]X$$

By stacking them together, we have

$$\begin{bmatrix} u_1 \times K[R_1, t_1] \\ u_2 \times K[R_2, t_2] \end{bmatrix} X = 0$$

To obtain X, we solve the null space of $\begin{bmatrix} u_1 \times K[R_1, t_1] \\ u_2 \times K[R_2, t_2] \end{bmatrix}$.

As lane markers are lines, it is not possible to match them directly between the image pair, and thus their locations cannot be triangulated in the same manner. Our solution is thus to first detect pixels which belong to lane markers on both images, then match them between the rectified image pair.

C. Lane Marker Triangulation

The process of lane marker triangulation is illustrated in Figure 4.

The strategy of lane marker detection is based on the color histogram and Canny edge detection [5] algorithm. For each image, we extract yellow or white pixels based on their

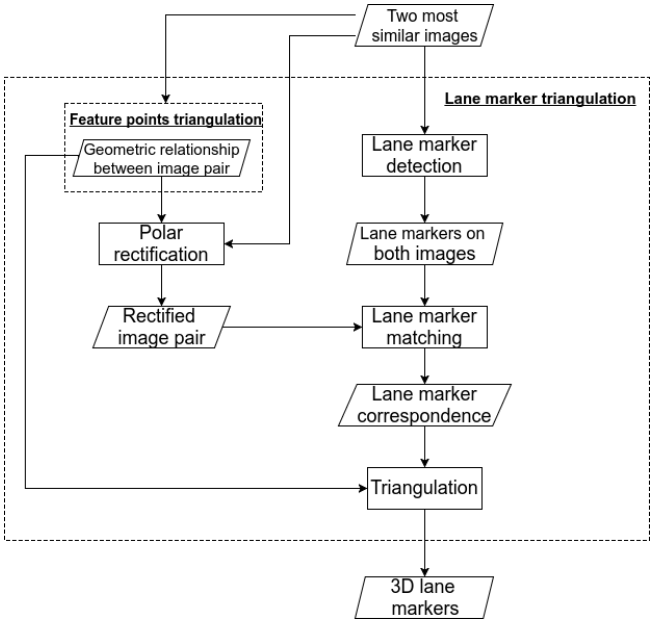


Fig. 4: Diagram depicting lane marker triangulation in SafeDrive.

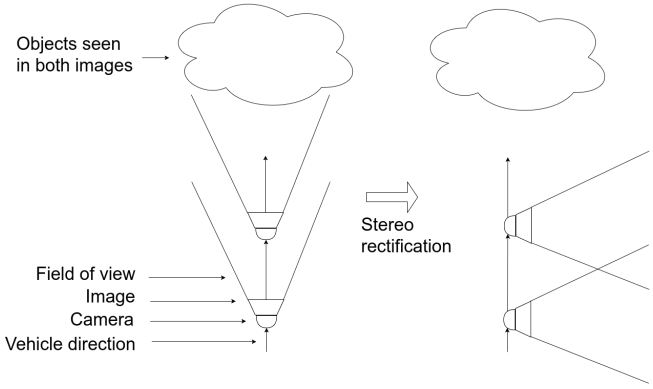


Fig. 5: Stereo rectification when camera is moving forward.

pixel value, since most lane markers in U.S. traffic system are of these two colors. To make it robust to illumination changes, we convert image from RGB to HSV color space. At the same time, we run Canny edge detection to extract the edges on the image. Finally, we take the intersection of yellow/white pixels and edges on the image.

We use the fundamental matrix computed in the previous step to rectify the image pair for triangulation. One important observation here is that vehicles are predominantly forward-moving, if we use classic stereo rectification [15], most of the information on the original image will disappear from the rectified image because camera is rotated almost $90°$, illustrated in Figure 5. To prevent this, we rectify the image pair in polar coordinates around epipole [16]. The vertical axis of the rectified image is the polar angle around epipole, and the horizontal axis is the distance to epipole. Polar rectification preserves every pixel from the original image, for camera movement in any arbitrary direction.

In order to match lane marker pixels on the rectified image, we first transform these lane marker pixels from the image pair to polar coordinates based on polar rectification. Consequently, each lane marker pixel from one image should have the same vertical coordinate as its corresponding pixel from the other image, because they share the same angle with respect to epipole. As a result, search space is reduced to one dimension only, which significantly speeds up the searching process. Furthermore, we do not need to check every pixel along the horizontal line, but check only those lane maker pixels that are likely to be the correspondence, which reduces the search space further. Currently, we compute a descriptor for each pixel, and compare it with the descriptors of those lane marker pixels in the other image that have the same vertical coordinates. After getting lane marker pixel correspondence on the polar coordinates, we transform the pixels back to Cartesian coordinate, and triangulate them using the same method as feature point triangulation. To further remove outliers, we reproject the triangulated 3D road marker to the original image pair and remove those road makers whose reprojection error exceeds a preset threshold.

D. Camera Registration

After building a 3D street model that consists of feature points and lane marker points, next step is to register current view with respect to the reconstructed 3D street model. We take the descriptors of each feature points from the database images, and match them against the feature descriptors from current view. After that, we have 3D to 2D correspondence, based on which, we solve Perspective-N-Point (PNP) problem [9] to get the relative pose of current camera with respect to the 3D model. We multiply the camera intrinsic parameters with the relative pose to get the projection matrix. Finally, we are able to project the lane markers in 3D model onto current view, so that driver is able to see the projected lane markers through the smartphone.**[we need to say up front why this projection is needed – maybe we already did, but need to remind the reader about this]** ★

IV. Experimental Evaluation

We have evaluated the performance and accuracy of SafeDrive using driving data acquired from a windshield-mounted smartphone. We recorded two driving datasets, one of which served as the database for alternate imagery, with its lane markers visible. Current view of the road comes from the other dataset, which is recorded at different times on different days under different driving (and resultantly, road) conditions. The current view is artificially corrupted by hand to simulate poor visibility of lane markers. Additionally, in the current view, direct lane detection is not attempted at all; this view is used only to construct a 3D model of the road and also to project the lanes onto this at the end.

The test case is located around coordinates $(44.979238N, 93.266568W)$. We have eight images in the database, as illustrated in Figure 6. The current view is selected from the other dataset around the same place, as shown in Figure 7.

Fig. 6: Images in the database around $(44.979238N, 93.266568W)$.

Fig. 7: Current view around $(44.979238N, 93.266568W)$. Note that the center lane markers have been removed from the current view for evaluation.

By matching features between the current view and all eight database images, the two images with the most feature matches are selected as the most similar image pair, which are shown in Figure 8a. Afterwards, lane markers are extracted from both images for later use (as described in Section III-C. We run feature matching between the images of this image pair (see Figure 8b), so that the feature correspondences can be used to calculate the fundamental matrix between them. For this test case, the fundamental matrix takes the following form:

$$\begin{bmatrix} 3.7989e-07 & -0.0005 & 0.2287 \\ 0.0005 & 1.3512e-06 & -0.2502 \\ -0.2294 & 0.2476 & 1.0 \end{bmatrix}$$

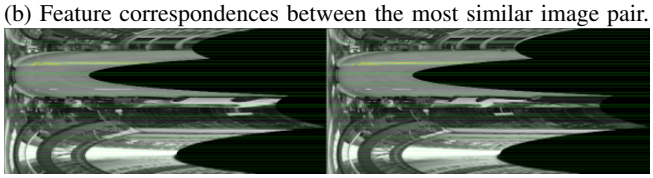
The fundamental matrix is further used to rectify the image pair. The rectified image pair are shown in Figure 8c. In this test case, since the vehicle was moving parallel to the lane marker, the lane marker is almost horizontal in the rectified image. This makes matching lane markers more challenging because there are many possible correspondences along the horizontal line. After matching lane marker pixels horizontally between the rectified image pair, as described in III-C, we get the lane marker correspondence as shown in Figure 8e. The outliers in Figure 8e will be removed later by reprojection.

Besides rectification, fundamental matrix is also used to recover the relative pose between the image pair. The intrinsic parameter of camera we use is:

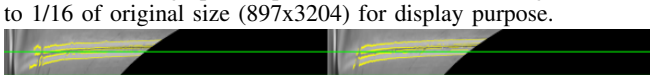
$$\begin{bmatrix} 1261.46807 & 0.0 & 619.89385 \\ 0.0 & 1259.44016 & 356.46599 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

(a) The most similar image pair in database found by feature-based matching, and lane markers detected on them.

(b) Feature correspondences between the most similar image pair.



(c) Rectified image pair in polar coordinates, with height reduced to 1/16 of original size (897x3204) for display purpose.



(d) Lane marker in the rectified image pair.

(e) Lane marker correspondences between the images of the chosen pair.

Fig. 8: The process of finding feature point and lane maker correspondences.

Consequently, the recovered rotation matrix is:

$$\begin{bmatrix} 0.999996 & 0.002211 & -0.001646 \\ -0.002212 & 0.999997 & -0.000380 \\ 0.001645 & 0.000383 & 0.999999 \end{bmatrix}$$

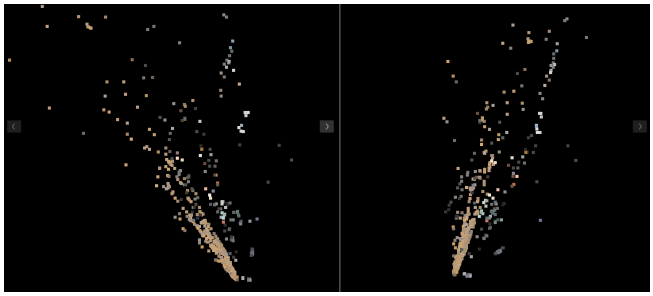
and translation vector is:

$$\begin{bmatrix} -0.069823 \\ 0.096280 \\ 0.992902 \end{bmatrix}$$

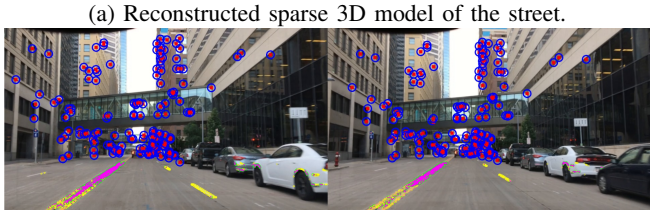
As can be seen. rotation matrix is close to identity, and translation vector is dominated by Z axis. This is because we were driving forward at that time, further strengthening the claim that cars do not usually have large lateral displacements when driving.

The rotation matrix and translation vector are used to triangulate both feature correspondences in Figure 8b and lane marker correspondences in Figure 8e. The result of sparse street reconstruction by triangulation is shown in Figure 9a. In order to measure the quality of both triangulations, we reproject the triangulated 3D points onto the original image

pair, and measure their distances to the 2D correspondence. The average reprojection error of feature points is 0.368289 pixel, and the average reprojection error of lane marker pixels is 0.897794 pixel, which does not include outliers whose reprojection error exceed a preset threshold, and are marked with yellow circles in Figure 9b.



(a) Reconstructed sparse 3D model of the street.



(b) Reprojecting 3D points onto the image pair. Blue circles: matched feature point; red markers: reprojected 3D feature point; green circles: matched lane marker; magenta markers: reprojected lane marker; yellow circles: lane marker with larger reprojection error.

Fig. 9: The final reconstructed 3D street model.

Next, we register the current camera into the sparse 3D street model. We first match the current view with the database images so that 3D street view is connected with the current view by 3D-2D correspondence, as illustrated in Figure 10a. Then we use the 3D-2D correspondence to solve for the current camera pose. We get the rotation matrix:

$$\begin{bmatrix} 0.999546 & -0.029875 & 0.004004 \\ 0.029743 & 0.999115 & 0.029731 \\ -0.004889 & -0.029598 & 0.999550 \end{bmatrix}$$

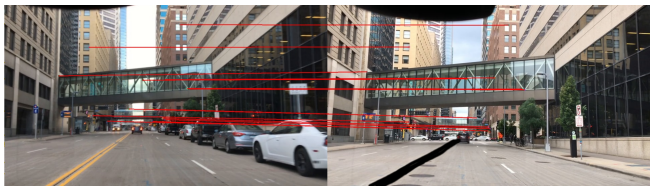
and translation vector:

$$\begin{bmatrix} 0.068230 \\ -0.261112 \\ -2.135684 \end{bmatrix}$$

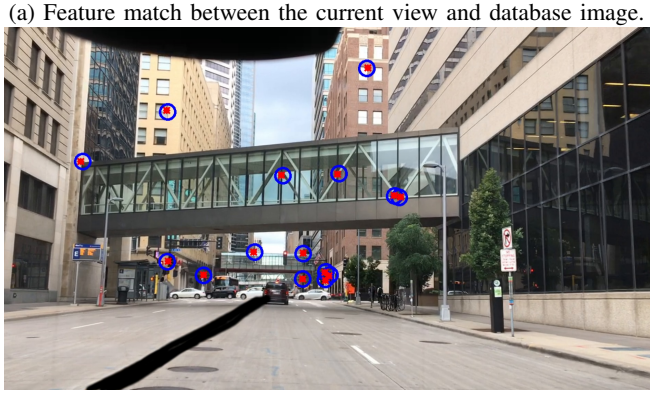
.

Using the rotation matrix and translation vector obtained for the current camera, we are able to project the lane markers from the 3D street view onto the current view. The result is shown in Figure 10c.

To compare the projected pixels with ground truth, we manually select a line to optimally represent the projected pixels, shown as a dashed line in Figure 11. At the same time, we select a line in the middle of lane marker as ground truth. We compare these two lines and find that the average offset between these two line segments is 4.803313 pixels, where the image size is 1280x720.



(a) Feature match between the current view and database image.



(b) Projecting 3D feature points onto current view. blue circle: matched feature point on current view; red marker: projected 3D feature point.

(c) Final result.

Fig. 10: Project lane markers onto current view

Fig. 11: Difference between the projected line (dash line) and ground truth (solid line).

To test the robustness of SafeDrive, we run it on several more test cases, as illustrated in Figure 12. One important note is that these test cases share the same parameters. This confirms that SafeDrive could run autonomously.

However, for cases in which the current camera is far

(a) Average offset: 2.375941 pixels.

(b) Average offset: 3.128704 pixels.

(c) Average offset: 4.581539 pixels.

(d) Average offset: 4.803313 pixels.

(e) Average offset: 6.631727 pixels.

(f) Average offset: 11.782807 pixels.

Fig. 12: Other test cases, image size of 1280x720

away from most 3D feature points in the street view, for an example of Figure 13a, or when 3D features are not equally spread, such as Figure 13b, SafeDrive is going to fail.

(a) Large offset(72.312232 pixels) when 3D features are far away from current camera.

(b) Large offset(36.181538 pixels) when most 3D features are on same side of the road.

Fig. 13: Fail test cases

In Figure 14, we investigate how the distribution of 3D feature points affect the accuracy of SafeDrive. As it shows, in order to make SafeDrive work, 3D feature points have to be distributed equally on both sides of the road, as well as widely spread. Therefore, SafeDrive works better in urban areas, while in rural areas, feature correspondences are neither enough or equally spread, causing SafeDrive to not work as well, such as in Figure 15.

V. Conclusions

We have presented an algorithm for visual lane detection and tracking under poor visibility conditions, and even in

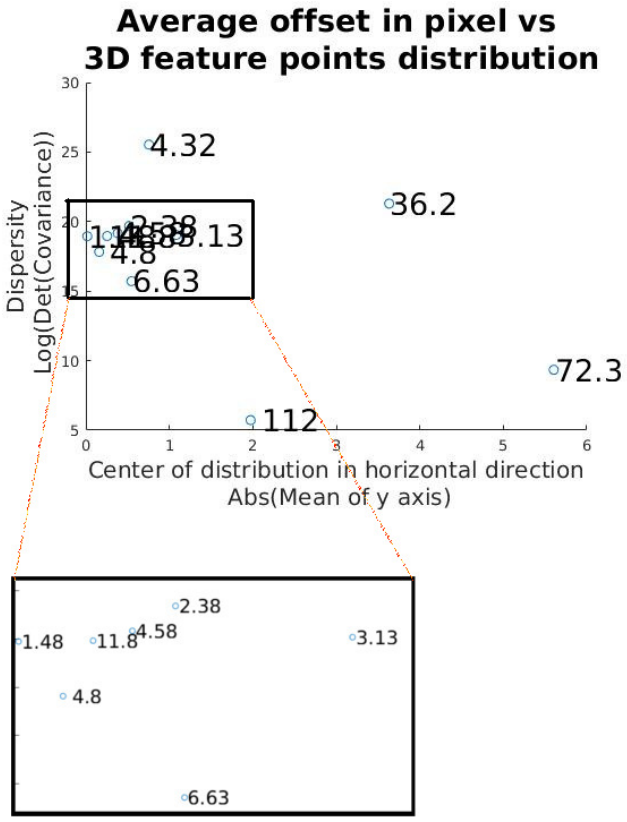
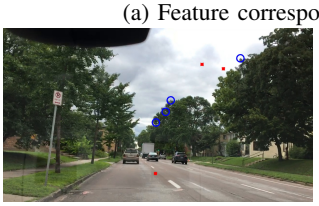


Fig. 14: Average offset against 3D feature points distribution.

(a) Feature correspondence on rural road.



(b) 3D feature points projected onto current view. blue circle: matched feature point on current view; red marker: projected 3D feature point.

(c) Rural test result: no visible projected lane marker, because projection matrix is totally wrong, indicated by the large feature projection error in Figure 15b.

Fig. 15: SafeDrive fails in rural test case

cases the road surface is barely visible. This approach leverages the availability of alternate imagery of the same location and the ability to perform lane tracking in such imagery, eventually mapping the lane detection back to the original camera image. With sufficiently robust visual lane-finding algorithms, accurate pose detection, and robust methods to relate the past image with the live frame,

we believe this algorithm can significantly improve driver safety. The ultimate goal for our work is to create an affordable system, and simultaneously improve the quality of autonomous transportation and occupant safety in road-going vehicles. Ongoing research is focusing on improved lane marker pixels matching, compressed data handling and optimization for enhanced performance, and extensive testing on data collected from a diverse set of geographic locations.

ACKNOWLEDGMENT

The authors are grateful to Michael Fulton for his assistance, particularly in developing the DriveData[2] Android™application and subsequent collection of a large volume of driving data.

REFERENCES

[1] Speeding-Related Traffic Fatalities by Road Type, Speed Limit, and State: 2009. Online, December 2012.

[2] Subaru EyeSight. Online, September 2015.

[3] Massimo Bertozzi and Alberto Broggi. GOLD: A parallel real-time stereo vision system for generic obstacle and lane detection. *Image Processing, IEEE Transactions on*, 7(1):62–81, 1998.

[4] Amol Borkar, Monson Hayes, and Mark T Smith. Robust lane detection and tracking with RANSAC and Kalman filter. In *ICIP*, pages 3261–3264, 2009.

[5] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[6] Kuo-Yu Chiu and Sheng-Fuu Lin. Lane detection using color-based segmentation. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 706–711. IEEE, 2005.

[7] Juan Pablo Gonzalez and Ümit Özgüner. Lane detection using histogram-based segmentation and decision trees. In *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*, pages 346–351. IEEE, 2000.

[8] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer, 1988.

[9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[10] Aharon Bar Hillel, Ronen Lerner, Dan Levi, and Guy Raz. Recent progress in road and lane detection: a survey. *Machine Vision and Applications*, 25(3):727–745, 2014.

[11] ZuWhan Kim. Robust lane detection and tracking in challenging scenarios. *Intelligent Transportation Systems, IEEE Transactions on*, 9(1):16–26, 2008.

[12] Karl Kluge and Sridhar Lakshmanan. A deformable-template approach to lane detection. In *Intelligent Vehicles' 95 Symposium., Proceedings of the*, pages 54–59. IEEE, 1995.

[13] Karl Kluge and Charles E Thorpe. Explicit models for robot road following. In *Vision and Navigation*, pages 25–38. Springer, 1990.

[14] Sergiu Nedevschi, Rolf Schmidt, Thorsten Graf, Radu Danescu, Dan Frentiu, Tiberiu Marita, Florin Oniga, and Ciprian Poco. 3D lane detection system based on stereo vision. In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 161–166. IEEE, 2004.

[15] Demetrios V Papadimitriou and Tim J Dennis. Epipolar line estimation and rectification for stereo image pairs. *IEEE transactions on image processing*, 5(4):672–676, 1996.

[16] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. A simple and efficient rectification method for general motion. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 496–501. IEEE, 1999.

[17] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: an efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[18] Arata Takahashi, Yoshiki Ninomiya, Mitsuhiko Ohta, MaKoto Nishida, and Munehiro Takayama. Rear view lane detection by wide angle camera. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 1, pages 148–153. IEEE, 2002.

[19] Chuck Thorpe and Takeo Kanade. Vision and Navigation for the Carnegie Mellon Navlab. In *Proceedings of the 1985 DARPA Image Understanding Workshop*, pages 143–152, 1985.

[20] Sebastian Thrun. Toward robotic cars. *Commun. ACM*, 53(4):99–106, April 2010.

[21] Qiong Wang, Jingyu Yang, Mingwu Ren, and Yujie Zheng. Driver fatigue detection: a survey. In *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, volume 2, pages 8587–8591. IEEE, 2006.

[22] Yue Wang, Dinggang Shen, and Eam Khwang Teoh. Lane detection using catmull-rom spline. In *IEEE International Conference on Intelligent Vehicles*, pages 51–57, 1998.

[23] Yue Wang, Eam Khwang Teoh, and Dinggang Shen. Lane detection and tracking using b-snake. *Image and Vision computing*, 22(4):269–280, 2004.

[2] https://github.com/fultonms/drivedata